The Representation and Solution of Problems in Applied Mathematics: an Artificial Intelligence Approach

THE REPRESENTATION AND SOLUTION OF PROBLEMS IN APPLIED MATHEMATICS: AN ARTIFICIAL INTELLIGENCE APPROACH

by George F. Luger, Alan Bundy and Martha Palmer
Department of Artificial Intelligence, University of Edinburgh, Scotland

A. Introduction

During the past ten years several groups of researchers in the field of Artificial Intelligence have addressed the issues arising in solving applied mathematics problems by computer. In Section B of this paper the work of four of the major projects in this area will be outlined. In Section C, the authors' work on the MECHO* project will be considered. Finally, some of the results of these projects and possible implications for educating engineers will be discussed.

B. Four Projects

B.1. Computer-Aided Circuit Analysis

Stallman and Sussman at MIT (1976) have designed and implemented a system for computer-aided circuit analysis. The system consists of a set of rules for electronic circuit analysis. This set of rules encodes physical laws such as Kirchoff's Law and Ohm's Law, as well as models of complex devices such as transistors. Facts, which may be given to or deduced by the system, represent information such as circuit topology, device parameters, voltages and currents.

The system works by forward reasoning. That is, the facts of the problem situation, combined with the rules encoding the physical laws that apply to this situation, drive the reasoning system. New deduced facts are tagged with justifications for deducing them, which include the problem facts and the inference rules used in their deduction. The justifications may then be examined by the user of the system to gain insight into the operation of the rule system as it applies to the problem. This is helpful for correction (debugging) of the rule system when it arrives at erroneous conclusions.

Furthermore, the justifications for new deductions are employed by the system in the analysis of fruitless search or blind alleys. This allows the system to avoid these situations in future reasoning.

The application of each rule in the system implements a one-step deduction. Four examples of these deductions, resulting from application of rules in the domain of resistive network analysis, are:

---

*. A Science Research Council Project, funded through the Department of Artificial Intelligence of the University of Edinburgh -- Dr. A. Bundy, grant holder, Dr. G. Luger and Mrs. M. Palmer assisting.

1. If the voltage on one terminal of a voltage source is given, then one can assign the voltage on the other terminal.

2. If the voltage on both terminals of a resistor is given and the resistance is known, then the current through the resistor can be assigned.

3. If the current through a resistor, the voltage on one of its terminals, ~~as well as~~ and the resistance of the ~~transistor~~ resistor are given, then the voltage on the other terminal can be assigned.

4. If all but one of the currents into a node are given, then the remaining current can be assigned.

Thus circuit-specific knowledge is represented by assertions in the data base and general knowledge about circuits is represented by laws or rules. Some laws represent knowledge as equalities, such as the laws for resistors stating that the current going into one terminal of the resistor must come out the other, or the laws for nodes stating that the currents must sum to zero. Other laws handle knowledge in the form of inequalities, such as the law that a diode can have a forward current if and only if it is ON, and can never have a backward current.

When a circuit-specific assertion (e.g., the voltage on a collector has values 3.4 volts) is added to the data base, several rules representing general circuit knowledge may match it and thus be activated (in the example, all the other elements terminals connected to the collector will be known to have 3.4 volts). The names of the activated rules will be put on a queue, together with information such as the place in the circuit that the rule is applied. Eventually this information will be taken from the queue and processed, perhaps making new deductions and starting the cycle over again.

When each general rule is processed it can do two useful things: make new assertions, or detect a contradiction. The new assertion, together with its antecedents, is entered into the data base. These antecedents, the asserting rule together with all the other rules asserted or used by the asserting rule, become useful when a contradiction is to be handled. This contradiction can arise when some previously-made arbitrary choice (for example, assuming some linear operating region for some non-linear component) was incorrect. The system then scans backward along the chains of deductions

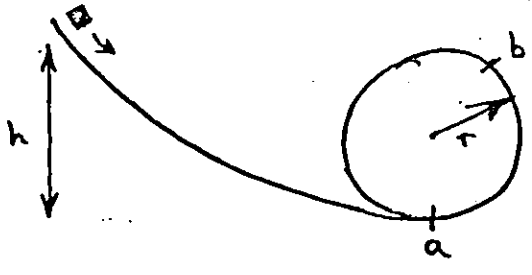
from the scene of the contradiction to find those choices that contributed to it. These choices are labelled NOGOOD and recorded in the system so that the same combination is not tried again. An example of a NOGOOD deduction could be one that says it cannot be simultaneously true that a transistor is cut off and a diode is conducting if the two are connected in series.

The forward reasoning together with the intelligent reduction of the possible search space effected by the NOGOOD assertions gives the system a flavor suggestive of the behaviour of the circuit expert. The justifications for deduced facts allow the user to examine the bases for their deduction. This is useful both for understanding the operation of the circuit, as well as for overcoming any problems arising within the set of general rules. For example, a device parameter not mentioned in the derivation of the value for a voltage has no part in determining that value. If some part of the circuit specification is changed (a device parameter or an imposed voltage or current) only those facts depending on the changed fact need be removed and rededuced, so small changes in the circuit may require only a small amount of new analysis.

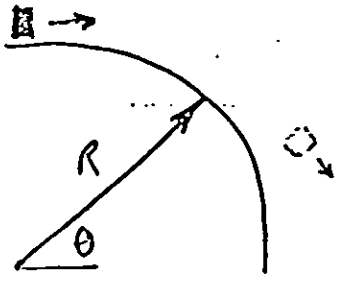
For more details of the work see Sussman et al., 1975, and Stallman et al., 1976.

B.2. Quantitative and Qualitative Reasoning

Also at MIT, de Kleer has written a computer program to solve problems involving the motion of a particle under gravity on a variety of paths (de Kleer, 1975). He calls these "roller coaster" problems. E.g.:



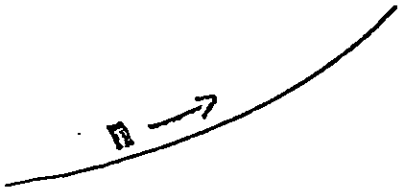
What is the minimum height h for which the particle will still loop the loop?



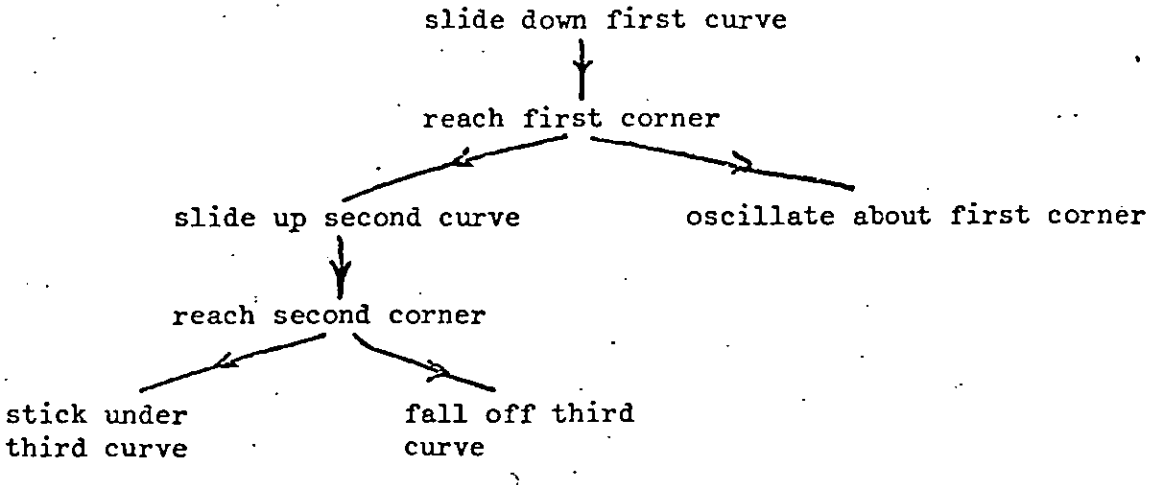
At what angle θ will the particle leave the circle?

These problems call for a mixture of qualitative and quantitative reasoning. The qualitative reasoning is responsible for deciding what kind of motion can take place, for instance in the loop-the-loop problem the particle might: oscillate about point a; fall off at some point b; or loop the loop. The quantitative reasoning is responsible for deciding precisely under what conditions each of these possibilities will occur. In de Kleer's program these two kinds of reasoning are clearly separated, with the qualitative reasoner proposing possibilties which are later checked out by the quantitative reasoner. This rigid separation eventually proves a liability since it hampers the flexible interaction of the two components.

The contribution of de Kleer's work lies in the design of the qualitative reasoner, which works by a process he calls "envisionment". For each shape of curve the program has a list of possible behaviours, e.g. a particle travelling uphill can reach the top and pass to the next



curve, or it can slide back down again. Each of these possible behaviours puts it in a new situation from which further possibilities arise. Thus the program builds up a tree of possible behaviours, for instance in the loop-the-loop example:



etc.

This tree is then passed to the quantitative reasoner which calculates what conditions have to hold for the particle to take the branches which lead to the desired state of looping the loop.

B.3. Reasoning in Semantically-Rich Domains

Two groups of researchers at Carnegie-Mellon University are studying reasoning patterns in areas of applied mathematics. Hinsley, Hayes, and Simon are studying the reasoning and solutions to algebra word problems, and Bhaskar and Simon the solutions to problems in chemical engineering thermodynamics.

These researchers describe problem-solving domains such as the above as semantically rich. This seems a good characterization in that large yet fairly well-defined amounts of prior semantic knowledge and task-related information are necessary for solving such problems. For example, it takes much more than an intelligent person and a "textbook" of relevant information to solve problems in thermodynamics. It is not information as available to the problem solver that is important, it is rather how the information is organized and stored, that is, information as useful.

As an example of a system without complete semantic information available, consider Bobrow's STUDENT. This system, designed to solve algebra word problems, attempts to solve these problems by a "direct translation" process which attempts to translate sentences of the problem directly into equations, and then to solve these equations. "The distance between Boston and New York is 250 miles" becomes "(the-distance-between Boston-and-New-York) = 250 x miles". STUDENT also recognizes key words such as "Distance" and can respond by adding "Distance = Rate x Time" to the equation list. This direct translation process and recognition of key words offers a good first approximation to human problem solving in these domains, but it is unable to deal effectively with the semantic information which is necessary to expose "The value of N nickels and D dimes is 93 cents" as nonsensical.

The study of the semantics of a problem domain is very important ~~both~~ for designing a computer program to solve problems, as well as for the human engineer solving problems. Several studies have shown (Marples,

1976; Marples and Simpson, 1975; and the authors' own work with problem-solving subjects, Luger, 1977) that it isn't what information is available to the problem-solving subject, but rather how this information is used, that brings success in problem solving. For example, knowing that a resolution-of-forces equation is relevant in determining accelerations of weights hanging over pulleys is only a small part of solving the problem. Much more important is the knowledge of how friction in a pulley may affect the tension in the string over the pulley, and how fixed contacts between the weights and string and the extensibility of the string may affect the acceleration of the particles and strings. This is the semantic content of the problem domain; it must be carefully specified for any computer program that would be of any interest, and it certainly marks the expertise of the successful problem solver.

Hinsley, Hayes, and Simon discuss the notion of problem schemata. These are sets of facts, relations and heuristics present in the problem-understanding process that allow the semantics of the problem domain to be properly processed. In the money example cited earlier, these facts and heuristcs would determine that nickels and dimes were non-divisible units of money worth five and ten cents respectively, and that no sum of them could equal 93 cents.

The Hinsley, Hayes, and Simon study ran five experiments to determine when and how human subjects employed problem-type schemata in problem solving, that is, how the humans organized and structured semantic information in the process of understanding and solving algebra word problems. In particular, they demonstrated how (1) subjects recognize problem categories; (2) that this categorization often occurs very early in reading the problem; (3) that subjects possess a body of information about each problem type ~~that~~ which is potentially useful for formulating problems of that type for solution; and (4) how this category information is actually used to formulate problems in the process of their solution.

The Bhaskar and Simon and Hinsley, Hayes and Simon research has not, as yet, led to their successful creation of a computer system to solve problems in different areas of applied mathematics. It is best to understand this work as a "prolegomena" to future problem-solving systems. This, indeed, is the main reason for including their work in

this survey -- not because it itself provides a useful model for machine or human problem solving, but because it provides a framework for future work in mechanical problem solving as well as an important key to the expertise and failings of human problem solvers.

The next important step in designing a mechanical problem-solving system is to specify the contents of the problem-type schema. That is, to select a problem domain and to attempt to fully specify the semantic information necessary to solve an interesting class of problems within this domain. The ISSAC system has done this for equilibrium problems (B.4) and the MECHO project has done it in the domain of pulley problems (C).

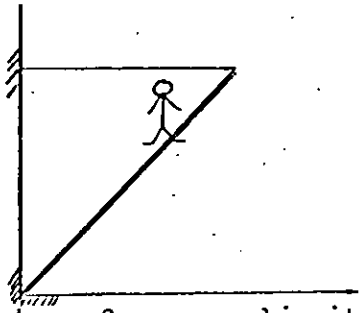
B.4. Solving Equilibrium Problems

Novak at the University of Texas, Austin, has developed a program called ISSAC for solving physics problems. ISSAC takes several simple statics problems stated in English, translates the English into several internal representations and solves the problem. Novak claims that it is necessary to use common sense knowledge and 'hidden' laws of physics to infer the relationships needed for solving the problem.

To investigate the Novak system, it is best to examine a problem in detail: "The foot of a ladder rests against a vertical wall and on a horizontal floor. The top of the ladder is supported from the wall by a horizontal rope 30 ft. long. the ladder is 50 ft long and weighs 100 lb, with its centre of gravity 20 ft. from the foot, and a 150-lb man is 10 ft. from the top. Determine the tension in the rope."

ISSAC uses syntactic and semantic information to parse the English sentences into a representation more amenable to automatic problem solving. Novak defined several categories in comparing every possible type of object that could be mentioned. A ladder is a PHYSICAL ENTITY, the top and the foot of a ladder are LOCATION PARTS, (meaning that the use of the word 'top' allows one to designate a particular area of the ladder), the weight and length of a ladder are ATTRIBUTES, a rung of a ladder is a PART, and 'by the wall' indicates a LOCATION for a physical entity. In the program the general categories are defined as SFRAMEs. Each SFRAME contains specific instructions about satisfactorily completing itself. For instance 'top' will trigger a LOCATION PART SFRAME, which will know that 'top' must be connected to a PHYSICAL ENTITY such as a 'ladder'. This information is very necessary to correctly associate all the 'tops' and 'feet' of ladders mentioned in the above problem.

It may seem painfully obvious that the 'top' in the second sentence and the 'top' in the third sentence refer to the same place and that the man is therefore 10 ft. from the point at which the rope is connected, but this is the type of inference that a program could easily fail to make, resulting in a misunderstood problem situation. When the parsing has been completed, all of these 'simple' inferences have been made, thanks to the SFRAMEs, and the program translates its abstract model of the ladder into more concise geometric form and presents it on a graphics screen. The picture produced is similar to the one below :



Producing a picture tests reference ambiguities such as the one mentioned above about the 'top'. Obviously, if the spatial relations cannot be worked out sensibly, something must have gone wrong in the parsing.

¶ The program is still not ready to generate equations. At this point the problem has only been understood in common sense terms of physical entities and their locations with respect to each other. Now the effects these physical entities have on each other need to be accounted for in terms of forces. This requires more specific physical information, such as the knowledge that because of gravity any object with a mass exerts a force downwards, and that any force exerted on an object causes a reaction of equal and opposite force to be exerted, assuming that the objects remain stationary. Using this information, ISSAC assumes forces exist everywhere two objects are in contact with each other. Again, these 'laws' may seem painfully obvious but realizing how carefully they need to be spelled out for the computer can give insights into possible problems students could have. ISSAC is also given another type of problem-solving information, this time relating to idealizations of real-world objects as they are commonly used in statics problems. ISSAC must recognize that the ladder can be idealized as a LEVER while the wall and floor are frictionless plane SURFACES and the man is a WEIGHT. ISSAC has been told that ladders are idealized as LEVERS.

Because of the limited domain, there is no reason for a ladder to be anything else, although it could easily be a WEIGHT in another type of problem. The man presents more of a problem, because even in this domain, men can be given more than one idealization, i.e. WEIGHT or PIVOT. To resolve this ambiguity, ISSAC makes use of the common sense knowledge that a WEIGHT is usually supported and a PIVOT usually supports something. In this problem the ladder supports the man, so the man must be a WEIGHT. It is clear that choosing an appropriate idealization is not always trivial, and that in complicated problem-solving areas it could present a serious deductive problem.

Since all of the problems ISSAC deals with are simple lever problems, once the forces have been identified generating equations is trivial. The sum of moments about a point must simply be set to zero. In mechanically writing equations for all moments, ISSAC generates several equations that a human problem solver would leave out. For instance, in the ladder problem ISSAC creates variables to represent certain horizontal forces exerted by the ladder, only to set those variables equal to zero in the next step. A competent problem solver should not need to go through such a step explicitly. However, Novak suggests that this is exactly the kind of unconscious leap that might confuse a poor student. In more complicated problem-solving situations, especially where motion is involved, taking note of all existing forces is only the first step. It is at this point that serious problem solving begins. Novak recommends that ISSAC, or a program with a similar approach, be extended to deal with dynamics problems. The authors have done this and discuss it in section C.

In summary, ISSAC solves twenty equilibrium problems competently. The program illustrates a sufficient semantic understanding of the problem situation to resolve referential ambiguities as in the 'top' example, and to interpret all objects and their relationships to each other correctly. In achieving such a level of understanding, certain necessary inferences are brought to light that can easily be overlooked in a classroom, and that might fill the gaps in a student's understanding.

C. The MECHO System

The MECHO project consists of writing a computer program to solve problems in applied mathematics. The scope of the project is broad: to take the English statement of a mechanics problem, give it to a computer, and receive in return answers to the questions asked in the problem. Three problem domains within the general area of mechanics have so far been considered: (1) acceleration, velocity, distance problems such as might arise with trains travelling between two stations (Bundy, Luger, Stone, and Welham, 1976); (2) the motion of particles over complex paths, such as the "roller coaster" problems tackled by de Kleer (Bundy, 1977); and (3) the domain of pulley systems (Luger, 1977). A simple problem in the third domain, in fact one of the first problems considered by the MECHO group, is:

> A man of 12 stone and a weight of 10 stone are connected by a light rope passing over a pulley. Find the acceleration of the man.

The thrust of the MECHO project research is pragmatic in that its primary goal is to design a computer program that can solve a wide class of problems. A further, but very important, goal of the project is the study of the running computer program as a model of human problem-solving activity. The trace of the program can be compared with the data of human protocols. The MECHO group has found this comparison fruitful, both as a source of new ideas which may be incorporated

into
~~in~~ the computer program itself, as well as to clarify important differences between the human and the mechanical problem-solving systems.

One of the important insights gained from studying human problem-solving protocols (Marples, 1975; Luger, 1977) has been to design the MECHO system as a forward or problem-driven and a backward or goal-driven problem-solving system. The remainder of this section will be spent clarifying this approach to problem solving and describing its implementation in the MECHO system.

Like the computer-aided circuit analysis described in section B1, the MECHO system employs problem-driven forward reasoning. This is accomplished by the creation and assertion of problem-type schemata. The word schema is used, following Bartlett and Piaget, to refer to a structuring of information, a loose confederation of relations which ~~that~~ represent the capacity to perform some task or function. In the terminology of Hinsley, Hayes, and Simon the problem-type schema contains the semantic information present in a problem situation, ~~along~~ together with the ability to use this information for solving a particular problem.

The problem type schema itself is composed of three parts: the declaration of entities, the set of facts and inference rules describing the problem situation, and a set of default facts and inferences. The declarations for the pulley problem above were often explicitly stated in the protocols we took of expert problem solvers: "We'll treat the man and weight both as particles, point masses .... I'll put these two dots on the paper and join them by this rope looped over a pulley". The entities declared in this situation are two particles, a pulley, and a rope over the pulley joining the two particles.

The set of facts and inferences relating these entities are similar to the following:

(a) an angle is assigned to the string between the pulley point and the left end

(b) an angle is assigned to the string between the pulley point and the right end

(c) fixed contact of particle1 to the left end of the string

(d) fixed contact of particle2 to the right end of the string

(e) the tension in the left section of the string is the same as the tension in the right end if the pulley is smooth (frictionless)

(f) the acceleration of the system is constant if the particles are in fixed contact with the string.

(a), (b), (c), and (d) above are examples of facts, and (e) and (f) are inferences that represent part of the semantic content of the pulley-system domain.

Finally, the pulley-system schema contains a set of default values. These values are facts and inferences such as:

(a) if the pulley is underspecified assume it to be smooth

(b) a rope is assumed to have constant length unless specified as elastic

(c) a rope has fixed contact with objects at its end points unless the problem states otherwise

(d) the pulley itself is fixed unless specified as movable.

The MECHO system's data base is ordered so that a problem-type schema, when it is invoked, is able to create new entities and assert new facts and inferences at the "top" of the data base. ~~and~~ It can also assert the default values at the "bottom". Thus when a call is made to the data base the facts and inferences about entities are checked first, ~~then~~ Finally, after every other check is made, the default values are assumed. In the pulley problem above, when a resolution-of-forces formula is attempting to assign a tension to the string, it will need to know whether the pulley is smooth. (If it is, a uniform tension will be assigned to the entire string.) When no information about the friction of the pulley can be found, as ~~is the case~~ in this problem, the default value of a pulley without friction will be asserted. Similarly, when the angle of the string is sought, the default value of the string with the weight hanging vertically downwards will be asserted.

So it is that the problem-type schema, representing the semantic information of the problem situation, is asserted. This represents the forward or problem-driven aspect of the MECHO problem solver. The goal-driven aspect is represented by the "Marples" algorithm for equation extraction.

The Marples algorithm was suggested by D. Marples from his work with engineering students at Cambridge (Marples, 1976). It is a procedure which starts from the desired unknown of the problem and works "backward", attempting to instantiate equations until a set of simultaneous equations sufficient to solve the problem is determined. The MECHO system has a focusing technique that "forces" the Marples algorithm to consider equations appropriate to the problem type, rather than to thrash about through lists of all possible equations. The focusing technique in the pulley system domain forces the Marples algorithm to consider first the general resolution-of-forces equations at the contact points of the string and weights.

Furthermore, the Marples algorithm is able to create "intermediate unknowns" in the process of solving the desired unknowns of the problems. In the pulley problem, for example, the desired unknown is the acceleration of the man. But it is impossible to determine the man's acceleration (using the resolution of forces) from the givens of the problem. Thus the Marples algorithm creates an intermediate unknown, the tension in the string at each end point of the string. When the inferences in the problem-type schema, (e) above, assign the same tension to each end of the string, the Marples algorithm produces the following equations:

$$T - 10g = 10a \quad \& \quad -(T - 12g) = 12a$$

These simultaneous equations are sufficient for solving the pulley problem.

The equations that may be applied to a problem situation are each encoded in a special format. The MECHO system, in asserting a particular equation, for example the resolution-of-forces equations above, specifies exactly the situations in which the equation is to be asserted. This includes the specification of each variable, the possible boundary values, and all semantic information necessary for the equation to be asserted.

The general resolution-of-forces equation, for example, sums all forces acting at a point. This would be trivially satisfied in the

problem above but would have to include the friction on a plane, the angle of the plane and other forces that could be acting in different pulley problems -- such as a pulley at the top of an inclined plane. Marples has stated (Marples, 1977) that the lack of this exact specification in using equations and misunderstanding the situation of their application is a major cause of mistakes for engineering students: it is not that the relevant equations are forgotten or ignored in problem solving, it is rather that the conditions of their use are misunderstood. The MECHO system specifies exactly each equation and the conditions under which it may be used.

Finally, the MECHO system includes automated procedures for solving sets of simultaneous equations. These are described in Bundy, 1975. As noted above, a more complete description of the MECHO system may be found in Bundy, 1977; Luger, 1977; and Bundy, Luger, Stone, and Welham, 1976; Stone, 1976.

D. Summary and Conclusions

This paper has attempted to summarize several Artificial Intelligence research projects in the areas of applied mathematics. These projects were intended to demonstrate the design and use of computerized systems both as interactive engineering aids and as models for solving problems in certain well-defined domains of applied mathematics. These systems may serve as models of how engineering problems may be understood and represented in the process of their solution -- both by man and machine.

Many of the same problems that arise for humans solving problems in applied mathematics are exactly those encountered by researchers attempting to design a computer system to solve problems: specifically, to design a system not limited to small sets of problems, but at the same time able to deal with the idiosyncracies of individual problems.

To deal with this conflict, each system creator has had to design an inferencing system. These include, for the computer-aided circuit analysis, the forward reasoning from the problem situation and the tagging of each

new fact with the conditions and rules used in its assertion. The de Kleer system attempts to control search by the quantitative vs. qualitative reasoning distinction and the use of envisionment. The Hinsley, Hayes and Simon and Baskar and Simon studies, while not explicitly constructing a problem-solving system, have outlined conditions that could apply to the creation of a successful problem solver. The Novak program solves twenty equilibrium problems from their English-language statement. It has sufficient semantic understanding to resolve all referential and relational ambiguities.

The MECHO system creates a forward or problem-driven and a backward or goal-driven inference system. The problem-driven aspect is represented by full specification of the problem-type schema. This includes the assertion of facts and inferences, both relating to the entities within the problem domain and sets of default values. The goal-driven aspect is represented by the Marples algorithm, which works backward from the desired unknown to the given facts of the problem. This often necessitates the creation of intermediate unknowns to link the desired unknown with the problem facts. The MECHO system also attempts to represent the semantics of each possible equation that may be asserted. This is intended to guarantee that the equation is only asserted in the manner and at the time appropriate.

The overall aim of this paper has been to provide a summary of some current work in artificial intelligence research in the areas of applied mathematics. This summary is meant not merely to make engineers aware of some interactive aids available (the computer-aided circuit analysis), but more importantly to give some idea of the representation of information and control of inferencing necessary for the successful problem solver in these domains. The reader is recommended to consult the references for more complete descriptions of each system surveyed.

Acknowledgments

The authors would like to especially thank David Marples of Cambridge for his continued encouragement in their work, the other members of the MECHO group for their support and ideas, and the SRC for financial assistance.

References

1. Bhaskar,R., & Simon, H.A. Problem solving in semantically-rich domains: An example from engineering thermodynamics, Cognitive Science 1, 2, 1977.

2. Bobrow, D. Natural language input for a computer problem solving system, Doctoral Dissertation, MIT, 1964.

3. Bundy, A., Luger, G.F., Stone, M. & Welham, R. MECHO: Year One, Report 22, Department of Artificial Intelligence, University of Edinburgh, 1976.

4. Bundy, A. Will it reach the top? Prediction in the mechanics world, Research Report 31, Department of Artificial Intelligence, University of Edinburgh, 1977.

5. Bundy, A. Analysing mathematical proofs (or reading between the lines), Proceedings of IJCAI4,Georgia, Cambridge, MIT-AI Press,1975.

6. de Kleer, J. Qualitative and quantitative knowledge in classical mechanics, MIT AI Lab.Report AI-TR-352, Cambridge, Mass., 1975.

7. Hinsley, D.A., Hayes, J.R., & Simon, H.A. From words to equations: meaning and representation in algebra word problems, C.I.P.331, Department of Psychology, Carnegie Mellon Univ., 1976.

8. Novak, G. Computer understanding of physics problems stated in natural language, TR NL30, Dept. of Computer Science, Univ. of Texas, Austin, 1976.

9. Luger, G.F. The use of protocols and the representation of semantic information in pulley problems, Research Report 36, Department of Artificial Intelligence, University of Edinburgh, 1977.

10. Marples, D. Final Report to U.K. Social Science Research Council of Cambridge Project, 1976.

11. Marples, D. & Simpson, P. Argument and technique in the solution of problems in mechanics and electricity, CUED/C - Educ/TRI, Dept. of Engineering Memo, University of Cambridge.

12. Stallman,R.M., and Sussman, G.J. Forward reasoning and dependency directed backtracking in a system for computer aided circuit analysis, MIT AI Tech. Report 380, September 1976.

13. Stone, Martha. PAT - pulleys and things, Working Paper 18, Department of Artificial Intelligence, Univ. of Edinburgh, 1976.

14. Warren, D. Epilog: users guide to DEC10 PROLOG, Department of Artificial Intelligence, Edinburgh, 1975.